Java Chapter 7: Object-Oriented Programming

Contents

Time required: 90 minutes

DRY

Don't **R**epeat **Y**ourself

Read: Think Java

- [Chapter 11 Designing Classes](#)

Do: Online Tutorials

- [Java OOP](#)

- [Java Classes/Objects](#)

- [Java Class Attributes](#)

- [Java Class Methods](#)

- [Java Constructors](#)

- [Java Encapsulation](#)

Tutorial 7.1 - Sammy the Shark

The following program demonstrates classes, objects, and methods.

An object is an instance of a class. We'll construct a **Shark** object called **sammy**.

Create a Java class named: **Shark.java**

Create a Java program named: **SharkMaker.java**

```java
/**
 * Name: Shark.java
 * Written by:
 * Written on:
 * Purpose: Class to create a shark
 */

public class Shark {
    // Define class data attributes
    private String name;
    private int age;

    // Define class getters and setters
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    // Define class methods
    public void swim() {
        // this references the class data attribute
        System.out.println(this.name + " is swimming.");
    }
}
```

```
 1  /**
 2   * Name: SharkMaker.java
 3   * Written by:
 4   * Written on:
 5   * Purpose: Demonstrate classes and objects
 6   */
 7
 8  public class SharkMaker {
 9      public static void main(String[] args) {
10          // Create a Shark object named sammy
11          System.out.println("Sammy the shark is created.");
12          Shark sammy = new Shark();
13
14          // Set the name and age
15          sammy.setName("Sammy");
16          sammy.setAge(3);
17
18          // Call an object method
19          sammy.swim();
20
21          // Get the object data attributes
22          System.out.println(sammy.getName() + " is " + sammy.getAge() + " years old.");
23      }
24  }
```

Example run:

```
Sammy the shark is created.
Sammy is swimming.
Sammy is 3 years old.
```

Tutorial 7.2 - Constructing Sharks

A constructor is a special method is used to create an object from a class. It is run as soon as an object of a class is instantiated.

Classes are useful because they allow us to create many similar objects based on the same blueprint. This program creates two objects from the same class using the default constructor and a parameterized constructor.

1. Create a Java class named: **Shark2.java**

2. Create a Java program named: **SharkMaker2.java**

3. Copy the previous tutorial code into each file.

```java
1  /**
2   * Name: Shark2.java
3   * Written by:
4   * Written on:
5   * Purpose: Class with parameterized constructor to create a shark
6   * If there is a parameterized constructor,
7   * a default constructor must be put in manually
8   */
9
10 public class Shark2 {
11    // Define class data attributes
12    private String name;
13    private int age;
14
15    // Default constructor
16    public Shark2() {
17    };
18
19    // Constructor with two parameters
20    public Shark2(String name, int age) {
21       this.name = name;
22       this.age = age;
23    }
24
25    // Define class getters and setters
26    public String getName() {
27       return name;
28    }
29
30    public void setName(String name) {
31       this.name = name;
32    }
33
34    public int getAge() {
35       return age;
36    }
37
38    public void setAge(int age) {
39       this.age = age;
40    }
41
42    // Define class methods
43    public void swim() {
44       // this references the class data attribute
45       System.out.println(this.name + " is swimming.");
46    }
47 }
```

```
 1  /**
 2   * Name: SharkMaker2.java
 3   * Written by:
 4   * Written on:
 5   * Purpose: Demonstrate classes and objects
 6   */
 7
 8  public class SharkMaker2 {
 9      public static void main(String[] args) {
10          // Create a Shark object named sammy
11          // Use the default contructor
12          System.out.println("\nSammy the shark uses the default constructor.");
13          Shark2 sammy = new Shark2();
14
15          // Set the name and age
16          sammy.setName("Sammy");
17          sammy.setAge(3);
18
19          // Call an object method
20          sammy.swim();
21
22          // Get the object data attributes
23          System.out.println(sammy.getName() + " is " +
24                  sammy.getAge() + " years old.");
25
26          // Use the parameterized constructor
27          System.out.println("\nSusie the shark uses the parameterized constructor.");
28          Shark2 susie = new Shark2("Susie", 2);
29          // Call the same methods as sammy
30          susie.swim();
31
32          // Get the object data attributes
33          System.out.println(susie.getName() + " is " +
34                  susie.getAge() + " years old.");
35      }
36  }
```

Example run:

```
Sammy the shark uses the default constructor.
Sammy is swimming.
Sammy is 3 years old.

Susie the shark uses the parameterized constructor.
Susie is swimming.
Susie is 2 years old.
```

Tutorial 7.3 - Coin Flip

The **Coin** class holds all the attributes and methods to flip a coin and return the value. The **CoinFlip** program creates a coin object and flips it 10 times.

Create a Java class named **Coin.java**

```java
1   /**
2    * Name: Coin.java
3    * Written by:
4    * Written on:
5    * Purpose: Class that flips a coin
6    */
7
8   // Import library for random numbers
9   import java.util.concurrent.ThreadLocalRandom;
10
11  public class Coin {
12      private final int MIN = 0;
13      private final int MAX = 1;
14      private int randomNumber;
15      private String sideUp;
16
17      public String Flip() {
18          // Generate a random number between
19          // MIN and MAX inclusive
20          this.randomNumber = ThreadLocalRandom.current().nextInt(
21                  MIN,
22                  MAX + 1);
23
24          // Determine which side was up
25          if (randomNumber == 0) {
26              this.sideUp = "Heads";
27          } else {
28              this.sideUp = "Tails";
29          }
30          // Return the side up
31          return this.sideUp;
32      }
33  }
```

Create a Java program that uses the Coin class named **CoinFlip.py** to flip a coin 10 times.

```java
1    /**
2     * Name: CoinFlip.java
3     * Written by:
4     * Written on:
5     * Purpose: Flip a coin object 10 times
6     */
7
8    public class CoinFlip {
         Run | Debug
9        public static void main(String[] args) {
10           String tossResult = "";
11           // Create a Coin object
12           Coin quarter = new Coin();
13
14           // Flip the coin 10 times
15           for (int i = 0; i < 10; i++) {
16               // Call object Flip Method, return a string
17               tossResult = quarter.Flip();
18               System.out.println(tossResult);
19           }
20       }
21   }
```

Example run:

```
Tails
Tails
Heads
Tails
Heads
Tails
Tails
Tails
Heads
Tails
```

Designing an Object-Oriented Program

The first step is to identify the classes the program will need.

1. Identify the real-world objects, the nouns.

   Customer
   Address
   Car
   labor charges
   Toyota

Some nouns contain other nouns. A Car can also be a Toyota. A customer would have an address. We need a Car and a Customer class.

2. Determine what describes or makes up the class, the data attributes.

   Customer: Name, Address, Phone

3. Determine the actions, the verbs, the methods.

   Move forward()
   Make purchase()
   create invoice()

   Customer: setName(), setAddress(), getAddress()

Keep everything that is relevant to the class in the class.

Assignment Submission

1. Attach the pseudocode.

2. Attach the program files.

3. Attach screenshots showing the successful operation of the program.

4. Submit in Blackboard.